

38

GONFLEZ VOTRE COLLECTION DE TURBOS

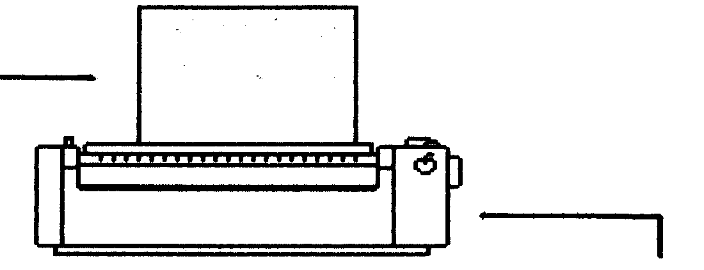
OCTOBRE 1987

TRANSFORMEZ LE FORTH 83-STANDARD DE LAXEN ET PERRY EN TURBO-FORTH

EDITORIAL

Après une période de marasme, JEDI semble s'être réactivé avec votre aide et le secours de nombreux articles. Ceux n'ayant pû être publiés dans ce numéro le seront dans les suivants. Pour vous mettre l'eau à la bouche: du FORTH (gestion graphique, variables locales, index trié de vocabulaire, gestion des attributs VIDEOTEX, graphisme avec carte HERCULES), du PROLOG (logique binaire en Turbo-PROLOG, simulation de commandes de robots) et des rubriques habituelles (logique, courrier le cas échéant) avec des questions et réponses.

JEDI change de ton: on continue à faire 20 pages si possible (excepté pour ce numéro-ci); une condition, écrivez des articles, envoyez des programmes, posez des questions, répondez aux questions. L'aspect dialogue sera renforcé. Mais le développement ne sera pas oublié: bientôt JEDI diffusera TURBO-Forth 83 Standard system pour MSDOS exclusivement. Ce nouveau FORTH ne traitera que des fichiers ASCII, sera plus compact et plus rapide que le F83 de Laxen et Perry tout en préservant sa syntaxe. Pour vous donner un avant goût et aligner votre F83 Laxen et Perry à la norme TURBO-Forth, vous trouverez dans ce numéro un programme permettant la digestion des programmes écrits au format ASCII. Notre intention est de coiffer BORLAND au poteau s'ils ont l'intention de diffuser eux aussi un TURBO-Forth. Avant il y avait les produits chers et les produits à prix BORLAND, maintenant, il y aura aussi le prix JEDI avec la qualité TURBO!



SOMMAIRE

NOTE:

C'est toujours le secrétaire qui réalise l'éditorial et la couverture. Mais si vous vous sentez inspiré pour exprimer votre humeur ou vos talents graphiques, n'hésitez pas, envoyez vos propositions, nous verrons ce que nous pourrons faire. Que ce soit à la manière de SAN-ANTONIO ou de VERLAINE avec votre plume trempée dans l'acide ou le miel, vous apporterez certainement un vent de fraîcheur à cette page.

Nos coordonnées: ASSOCIATION JEDI 17, rue de la Lancette 75012 PARIS
tel président: (1) 43.40.96.53
tel secrétaire: (1) 46.56.33.67

Si tous les langages permettent une sauvegarde des programmes source en ASCII, le langage FORTH faisait bande à part avec une gestion de bloc bien exotique. Voici un utilitaire permettant de combler cette lacune.

> TURBO-FORTH (NR)
> un produit NOTBORLAND (NTM)
>
> NR=Not Registered
> NTM=Not Trade Mark

par Marc PETREMANN
F83 Laxen et PERRY version MSDOS

C'est bien simple, depuis que P.KAHN, directeur de BORLAND USA s'est expatrié aux States, nous sommes envahis (en bien, merci Philippe...) par Turbo-PASCAL, Turbo-C, Turbo-PROLOG, Turbo-BASIC... et caetera.

Il n'y a donc pas de raison que nous ne nous fendions pas d'un programme permettant de transformer le FORTH 83-Standard de Laxen et Perry en une version plus conviviale et conforme à l'esprit de BORLAND, je cite: Turbo-FORTH.

Si le titre de cette rubrique fait un peu gag, le programme proposé ci-après n'en est pas moins intéressant. Rappelons l'esprit qui gouverne la gamme Turbo-xxx:

- un programme est traité sous forme de fichier ASCII à partir d'un éditeur de texte plein écran.

- le programme est compilé à partir du compilateur.

Or, avec FORTH nous disposons du compilateur. L'éditeur plein écran peut avantageusement être remplacé par un programme de traitement de texte. Pour ma part, WORDPERFECT convient très bien à cette tâche. En effet, il est possible de quitter le traitement de texte, lancer FORTH, compiler le programme et revenir au programme source:

DEPUIS WORDPERFECT:
- taper son fichier ASCII contenant un programme FORTH
- sauvegarder sous format ASCII en activant CTRL-F5 et option 1; confirmer par O si le fichier existe déjà sur le disque
- après sauvegarde, activer CTRL-F1, et passer sous DOS avec l'option 1

SOUS DOS:
- taper F83, cette version incluant le programme ci-après en version compilée

SOUS FORTH:
- taper INCLUDE [d:]fichier.ext (paramètre d: facultatif), exemple: INCLUDE GRAPHIC.TXT
- exécuter le programme
- sortir de FORTH en tapant BYE

SOUS DOS
- taper EXIT

RETOUR A WORDPERFECT
- vous êtes de retour à WORDPERFECT, votre fichier source est toujours résident et le curseur à l'endroit précis où vous l'avez laissé avant de passer sous DOS.

EN PRATIQUE

Vous allez souffrir encore un peu si manipuler des blocs vous rebute réellement. Entrez dans un fichier FORTH nommé INCLUDE.BLK le listing ci-après:

```
10 CREATE-FILE INCLUDE.BLK
OPEN INCLUDE.BLK
1 EDIT
0 NEW
    puis entrez le listing
```

Votre programme occupe n blocs; quittez l'éditeur en tapant DONE puis 1 n THRU où n est le numéro du dernier bloc édité. Exemple, votre programme occupe 5 blocs (bloc 1 à 6), compilez-le en tapant 1 6 THRU.
La sauvegarde en version compilée est exécutée par les commandes:

```
HERE FENCE !
SAVE-SYSTEM TURBOF83.COM
BYE
```

Vérifiez le bon fonctionnement de TURBO-FORTH en tapant TURBOF83.

Essayez un premier fichier texte; revenez sous DOS et tapez:

```
COPY CON: BOUCLE.TXT
: BOUCLE
   20 0 DO CR I LOOP ;
BOUCLE
^Z
```

Vérifiez le contenu du fichier en tapant TYPE BOUCLE.TXT.

Testez TURBOF83 en action en tapant:

```
 TURBOF83
et sous FORTH
 INCLUDE BOUCLE.TXT
```

Le programme est compilé puis exécuté, car BOUCLE figure en mode exécution après la compilation de sa définition.

COMMENT CA MARCHE

Là vous vous dites: c'est dément, c'est super... vous piaffez d'impatience d'essayer, vous faites baver votre chien en oubliant de lui donner à manger (pauvre bête) et qu'il lorgne sur vos frites que vous avez laissé refroidir, captivé par le contenu de cet article que vous êtes...

La première partie est toute simple. On définit une série de vecteurs destinés à initialiser une table CONTROL-TABLE se substituant à CC-FORTH

La seconde partie, définit trois mots en assembleur FORTH, (OPEN-F), (CLOSE-F) et F-GET. Le mot F-GET est capital; il dépose à chaque exécution une valeur correspondant à un caractère lu séqentiellemnt dans le fichier ASCII ouvert. La vectorisation ultérieure de F-GET dans KEY fait croire au système FORTH que tout caractère lu dans un fichier provient d'une frappe au clavier.

La troisième partie gère les cas d'erreur et de fin de fichier.

La dernière partie définit INCLUDE. Rien à dire si ce n'est le comportement particulier de ce mot.

INCLUDE ouvre un fichier, compile ou exécute son contenu si c'est du code FORTH puis referme le fichier. Il n'y a pas de collision à compiler un fichier ASCII depuis un bloc FORTH. Par contre, on ne peut exécuter INCLUDE depuis un fichier ASCII. Si vous voulez fusionner plusieurs fichier, définissez un bloc contenant:

```
INCLUDE fichier1.txt
INCLUDE fichier2.txt
...
INCLUDE fichiern.txt
```

ou fusionnez vos fichiers à l'aide de votre traitement de texte.

La compilation à l'aide d'INCLUDE ne génère pas d'en-tête pour le fichier dans le dictionnaire contrairement à OPEN.

La longueur des lignes du fichier ASCII est limitée à 80 caractères, une tabulation est comptée pour un caractère (un espace).

L'écho à l'affichage lors de l'exécution de INCLUDE permet

de repérer une éventuelle erreur.

Le fichier ASCII peut avoir une taille quelconque.

ATTENTION AUX INCOHERENCES: Si vous débutez en FORTH et que vous reproduisez un programme présenté sous forme de blocs en fichier ASCII, ne reprenez pas les mots suivants:

```
 -->
 THRU
 +THRU
```

Utiliser avec prudence ceux-ci:

```
 LOAD
 FROM
 OPEN
```

LE LISTING

```
\ Chargement de fichiers texte
DEFER "A DEFER "B DEFER "C DEFER "D DEFER "E
DEFER "F DEFER "G DEFER "I DEFER "J DEFER "K
DEFER "L DEFER "N DEFER "O DEFER "Q DEFER "R
DEFER "S DEFER "T DEFER "V DEFER "W DEFER "Y
DEFER "Z  DEFER "0

' NOOP IS "0       ' (CHAR) IS "A    ' (CHAR) IS "B
' (CHAR) IS "C     ' (CHAR) IS "D    ' (CHAR) IS "E
' (CHAR) IS "F     ' (CHAR) IS "G    ' (CHAR) IS "I
' (CHAR) IS "J     ' (CHAR) IS "K    ' (CHAR) IS "L
' (CHAR) IS "N     ' (CHAR) IS "O    ' (CHAR) IS "Q
' (CHAR) IS "R     ' (CHAR) IS "S    ' (CHAR) IS "T
' (CHAR) IS "W     ' (CHAR) IS "Y    ' (CHAR) IS "Z

CREATE CONTROL-TABLE     CONTROL-TABLE CC !
] 0      "A     "B     "C     "D     "E     "F
"G     BS-IN   "I     "J     "K     "L     CR-IN
"N     "O     P-IN   "Q     "R     "S     "T
BACK-UP  "V   "W   BACK-UP   "Y  "Z      CHAR
CHAR   CHAR    CHAR     CHAR [

\ Chargement de fichiers texte
HEX
CODE (OPEN-F)  ( adr-nom-fichier --- hndl fl)
  DX POP   3D02 # AX MOV   21 INT   AX PUSH   U<
  IF    0 # AX MOV
  ELSE  1 # AX MOV    THEN
  1PUSH END-CODE
CODE (CLOSE-F)  ( hndl ---)
  BX POP 3E # AH MOV 21 INT NEXT END-CODE

VARIABLE F-HANDLE
VARIABLE K-BUF

LABEL F-ERROR  0 # AX MOV 1PUSH
CODE F-GET   ( --- n )
  F-HANDLE #) BX MOV   1 # CX MOV   K-BUF # DX MOV
  3F # AH MOV   21 INT   F-ERROR JB   CX AX SUB   0<>
  IF     1A # AL MOV
  ELSE   K-BUF #) AX MOV    THEN
  AH AH SUB   1PUSH    END-CODE
DECIMAL

\ Chargement de fichiers texte
VARIABLE F-NAME  15 ALLOT
: (GET-FNAME)
  14 MIN DUP  ROT ROT
  F-NAME SWAP MOVE F-NAME + 0 SWAP C!  ;

: GET-FNAME
  BL WORD COUNT (GET-FNAME) ;

: EOF
  ['] (KEY) IS KEY
  ['] (CHAR) IS "I
  ['] (CHAR) IS "J
  ['] NOOP IS "0
  ['] RES-IN IS "Z
  ['] (?ERROR) IS ?ERROR
  F-HANDLE @ (CLOSE-F) ;

: CONTROL-Z
  ." Fin de fichier " CR EOF BACK-UP CR ;

: ?ERR-0
   DUP IF    EOF (?ERROR)
        ELSE DROP 2DROP    THEN ;

\ Chargement de fichiers texte
: TAB-IN
  DROP 32 (CHAR) ;

: INCLUDE
  GET-FNAME F-NAME  (OPEN-F)
  IF     F-HANDLE  !
         ['] TAB-IN IS "I
         ['] DROP IS "J
         ['] F-GET IS KEY
         ['] EOF IS "0
         ['] CONTROL-Z IS "Z
         ['] ?ERR-0 IS ?ERROR
  ELSE   TRUE ABORT" Fichier non trouvé"
  THEN ;
```

EN CONCLUSION

Nous espérons avoir fait des heureux en leur simplifiant la manipulation des fichiers FORTH.

Ce qu'il manque maintenant, c'est peut-être un véritable éditeur de fichiers ASCII travaillant en mode plein écran.

Les fonctions assembleur utilisées dans ce programme ont été inspirées d'un article paru dans FORTH DIMENSION.

Voilà, occupez-vous de votre chien, il est en train de dévorer votre bas de pantalon, perdant patience à attendre sa paté.

Ndlr: Si P.KAHN sort un jour un vrai Turbo-FORTH, qu'il ne nous en veuille pas d'avoir usurpé un titre peut-être déjà déposé. S'il est sympa, qu'il nous envoie un échantillon, on n'attend que ça pour en dire du bien... ■

DERNIERE MINUTE: nous apprenons que les logiciels POSTSCRIPT et RAPIDFILE ont été écrit en FORTH. Voici qui nous conforte dans l'idée que FORTH est un langage intéressant et d'avenir.

Concernant RAPIDFILE, ce logiciel de traitement de base de données, réalisé par ASHTON TATE est le dernier né d'une série prestigieuse:

- dBASE II (écrit en BASIC et compilé)
- dBASE III (écrit en C)

COURRIER: Cher Secrétaire

Je regrette vivement de n'avoir pu me rendre à votre convocation d'assemblée.

J'ai reçu celle-ci le lendemain au courrier et 00 km nous séparent.

Je désire toutefois témoigner de l'intérêt pour JEDI tel que vous l'animez et de ma reconnaisance profonde à votre égard.

Vous vous alarmez du peu d'activité des membres en été! Hélas, par temps de canicule, la baignade inspire plus que le clavier. Je ne suis FORTHa qu'à mes heures, en principe l'hiver.

Vous portez l'animation de cette asociation au plus haut qu'il est possible, je vous en suis bien reconnaissant, mais la participation de chacun de nous est

bien modeste, pour ne pas dire inexistante. Peut-être la télématique devrait-elle apporter le dialogue permament qui fait défaut me semble-t-il entre chacun d'entre nous.

Une machine connectée sur une ligne tléphonique suivant un calendrier servirait de boîte aux lettres globale.

L'animation viendrait ainsi de partout et les préoccupations ou les suggestions pourraient être reprises et traitées par chacun. De cette façon, des problèmes mobilisateurs pourraient être sélectionnés, des groupes constitués, des faux problèmes éliminés, etc...

Mais la mise en place des moyens est peut-être prohibitive pour des préoccupations actuelles. L'association ne se développe pas, écrivez-nous, bien heureux, car à coté l'informatique familiale s'effondre, la consommation intérieure ne s'est pas dveloppée!... faute d'imagination et de communication.

Je renouvelle mon espoir de voir l'Association poursuivre son action, si précieuse pour moi.

Forthement vôtre

J.CHANDRU
26130 ST PAUL TROIS CHATEAUX

REPONSE: Cher adhérent,

Votre courrier me fait bien plaisir, et venant de la part d'un membre ayant déjà participé activement au contenu de la revue, nous encourage à continuer (nous: SECRETAIRE, TRESORIER, PRESIDENT).

Peut-être l'été a-t-il été trop chaud. Je souhaite en tout cas qu'après avoir tiré l'alarme, les choses bougent un peu.

Nous avons fait une première expérience télématique en proposant un service FORUM sur (3615)SAM. C'etait un échec; moins de une minute de communication par semaine entre juin 86 et septembre 86. De ce fait, les logiciels téléchargeables ont été supprimés. Reste la messagerie très agonisante faute d'animation (peut-être défunte à ce jour).

D'accord pour toute idée de serveur télématique, mais ne recommençons pas les mêmes erreurs. Lancer un serveur, même monovoie en réseau commuté, sans proposer au préalable une liste déjà fournie de services, c'est courrir à nouveau au désastre. De plus, les communications téléphoniques coûtent relativement cher. Pour ne faire aucune discrimination entre adhérents de province et adhérents de PARIS, il semble obligatoire de recourir au 3615. Mais un serveur exclusivement JEDI coûte les yeux de la tête.

Au mieux, JEDI pourrait se faire héberger par un serveur déjà opérationnel ou un prestataire de service. Mais comme je ne puis tout faire, si des adhérents ont des relations ou des contacts dans le domaine de la télématique, nous leur laissons carte blanche pour mener à bien l'idée d'un service de mesagerie JEDI. Inutile d'habiter à PARIS pour monter ce service.

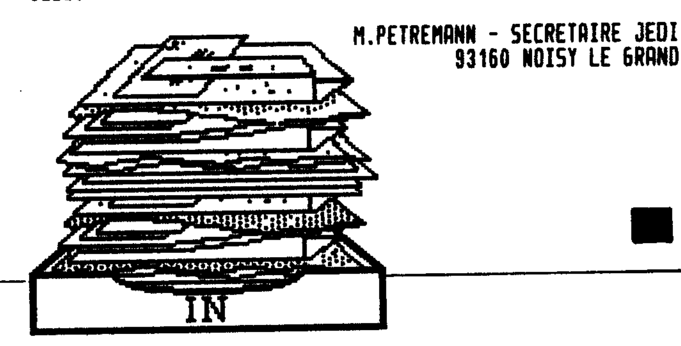
Concernant l'effondrement de la micro familiale, c'était prévisible. On a tout fait pour écoeurer les utilisateurs:

- sortir un SPECTRUM SINCLAIR 64K sans CP/M
- sortir un TO7/70 puis un TO9 sans FLEX
- sortir un ORIC équipé d'un 6502 sans donner aucun moyen de digérer les programmes APPLE...
- et toutes les machines tel ALICE, NEWBRAIN, MPFII, et série MSX dont la vente a été sans suite, voire une véritable escroquerie...
- pondre des SQUALE avec un service après vente inexistant...
- sortir 200 jeux sur un même modèle de machine avec le même scénario (tirer sur des martiens, des vénusiens, des GFRETGJJ...) contre un utilitaire tenant à peine la route (700 Fr un traitement de texte de 2k sur cassette pour NewBRAIN, merci, on a donné...).

Certes, il y a eu aussi de bonnes choses, mais raisonnons logiquement, et comprenons qu'un utilisateur ne peut plus investir 5000 Fr ou plus sans avoir la certitude d'avoir fait le bon choix. C'est pourquoi, il ne reste plus en concurrence sur le marché du hobby informatique que les systèmes dits COMPATIBLES PC et les systèmes à base de µP 68000 (ATARI, COMMODORE, MacINTOSH). Les autres, à la trappe...

Ceci dit, il reste encore beaucoup à inventer et la créativité ne mourra pas, nous espérons en faire la preuve encore longtemps et avec l'aide de tous les membres de JEDI.

M.PETREMANN - SECRETAIRE JEDI
93160 NOISY LE GRAND



Cher secrétaire,

Comme je serai désolé de ne plus recevoir votre "remarquable quasi mensuelle" publication, je vous envoie un peu de matière qui, je l'espère, vous sera utile.

En espérant que ma modeste contribution sera utile, je vous prie de recevoir mes meilleurs sentiments

Joël STEIN - MAROLLES EN BRIE

Je vous propose un source pour assembleur 8086 normalement inclus dans un source FORTH (fabrication maison à partir du FORTH MVP), qui, au risque d'encourir les foudres des puristes, permet le calcul flottant, ce qui, en FORTH, est tout de même souvent utile.

Bien entendu la forme un peu lourde d'écriture peut être avantageusement remplacée par des écrans forth normaux, et on peut utiliser l'assembleur forth pour obtenir les mêmes résultats. Je laisse ce soin aux amateurs de ce genres d'exercices.

La plupart des routines de convertion ascii\flottant et flottant\ascii sont dérivées du BLUEBOOK of ASSEMBLY ROUTINES for the IBM PC & XT de Christopher L.Morgan. Les routines de calcul à proprement parler sont issues d'une longue compilation de différents documents extraits de revues (BYTE, DR DOBB'S, COMPUTER LANGAGE, etc...) et de mon expérience personnelle. Le choix du format est un compromis entre la précision et la vitesse d'exécution.Il ne permet pas la compatibilité avec le coprocesseur 8087, cependant le source est aisément modifiable dans ce sens, mais dans ce cas sans doute vaut-il mieux utiliser un autre système.

Ce flottant est donc constitué de 3 mots de 16 bits. Les 8 bits de poid le plus fort représentent l'exposant binaire signé, les 40 bits suivants étant la mantisse signée normalisée sous la forme x.xxxxxxxxxxx x étant un chiffre de 1 à 9 (cette normalisation n'est pas standard). Exemple:

928.345 est représenté sous la forme 9.28345E+2
0.0928 est représenté sous la forme 9.28E-2

GLOSSAIRE

Les mots FORTH permettant la conversion s'utilisent comme suit:

ASC>FP (Add Count ---- n1 n2 n3)
Add = adresse de la chaine de caractère représentant le nombre
Count = longueur de la chaine
n1 , n2 et n3 représentent les 3 fois 16 bits du flottant

FP>ASC (n1 n2 n3 --- Add)
n1,n2 et n3 sont les trois cellules du flottant
Add pointe sur le count de la chaine de caractères représentant le nombre

F+ (FP1 FP2 --- FP3)
FPx représente un nombre flottant donc 3 cellules (n1 n2 n3) FP3 = FP1+FP2

F- (FP1 FP2 ---- FP3)
FP3 = FP1 - FP2

F* (FP1 FP2 ---- FP3)
FP3 = FP1 * FP2

F/ (FP1 FP2 ---- FP3)
FP3 = FP1 /FP2

FDUP (FP1 --- FP1 FP1)

FSWAP (FP1 FP2 --- FP2 FP1)

F0 constante qui retourne 0 0 0 (0 flottant)

INT (FP1 --- n)
convertit un flottant en entier signé (-8000H =< n =< 7FFFH)

FLOAT (n ---- FP)
convertit un entier signé en flottant

FOVER (FP1 FP2 --- FP1 FP2 FP1)

F@ (Add --- FP)
extrait les trois cellules constitant FP de Add

F! (FP Add ----)
range les trois cellules constituant FP dans Add

Comme on peut le constater dans le source les erreurs (ERRFP) ne sont pas traitées, et le sous programme se contente de faire un RET. avis aux amateurs !!!!!

ATTENTION comme les erreurs ne sont pas traitées, pour éviter l'erreur fatale si redoutable, la division par 0 retourne 0 (je sais, c'est très mal mais là encore le temps me manque et si quelqu'un veut bien se donner la peine, je suis preneur !!!!)

En attendant la suite je vous souhaite à tous de bien vous amuser...

LISTING

```
; DEBUT DES ROUTINES FLOTTANTS
; CES ROUTINES SONT APPELEES PAR FORTH
;
;
;  DEFINITIONS DES ZONES RESERVEES
; -------------------------------
FPTEMP1     DB   13 DUP (0)  ; FORMAT INTERNE DE CALCUL
FPTEMP2     DB   13 DUP (0)  ; FORMAT INTERNE DE SAISIE
DECBUFF     DB   25 DUP (0) ; BUFFER DE CONVERSION BIN/DEC
DECSIGN     DB   0 ; SIGNE DECIMAL
DECEXP DW   0   ; EXPOSANT DECIMAL
DECFLAG     DW   0 ;
BUF80B DB   20 DUP (0)
BUFCARIN    DB 25 DUP (0)  ; BUFFER DE SAISIE
NUMB   DB   8 DUP (0)
BUFCAROUT   DB 26 DUP (0)
FPERR  DB   0
TBUFF  DB   5  DUP (0)
SGNOPER1    DB   0
SGNOPER2    DB   0
MULEXP1     DB   0
MULEXP2     DB   0
; ZONE DE TRAVAIL
TEMPDIV     DB   16 DUP (0)
OPER1  DB   9  DUP (?)
OPER2  DB   9  DUP (?)
RESMUL      DB   14  DUP (?)
RESMANT     DB   9 DUP (0)
QUOTIENT    DB   8 DUP (0)
;
;
;      FQUOTIENT
       DB   89H
       DB   'FQUOTIEN'
       DB   'T'+80H
       DW   RNO-6      ; a remplacer
FQOT        DW
       DW   QUOTIENT
;
;      FACC
       DB   85H
       DB   'FPAC'
       DB   'C'+80H
       DW   FQOT-0CH
FPACC       DW   DOCON
       DW   OPER1
;
;      FPOP
```

```
        DB   84H
    DB  'FPO'
        DB   'P'+80H
        DW   FPACC-8
FPOP    DW   DOCON
        DW   OPER2
;
;
;       BUFSORT
    DB  87H
    DB  'BUFSOR'
    DB  'T'+80H
    DW  FPOP-7
BUFSO   DW   DOCON
    DW  BUFCAROUT
;
;
;       RESULT
    DB  85H
    DB  'FPRE'
    DB  'S'+80H
    DW  BUFSO-0AH
FPRES   DW   DOCON
    DW  NUMB
;
;
;       FPENTREE
    DB  84H
    DB  'FPI'
    DB  'N'+80H
    DW  FPRES-8
FPINP   DW   DOCON
    DW  BUFCARIN
;
;
;
;       ERRFP
ERRFP:  RET
            FPLEC
;   LE NOMBRE DOIT ETRE DANS BUFCARIN EN ASCII
;   LE PREMIER OCTET CONTIENT LE COUNT
LECFP:  PUSH DI
    PUSH DX
    PUSH CX
    LEA  DI,BUFCARIN+2
    MOV  CX,WORD PTR BUFCARIN
    CMP  CL,CH
    JZ   FPLEC1
        MOV  DL,CH
        SUB  DH,DH
        ADD  DI,DX
    MOV  AL,BYTE PTR [DI]
    INC  BYTE PTR BUFCARIN+1
    JMP  FPLEC2
FPLEC1: MOV  AX,0
FPLEC2: POP  CX
    POP  DX
    POP  DI
    RET
;               CONVERSION ASCII SIGN EN BINAIRE
SGNDEC16IN:     MOV  DX,0
        MOV  CH,0
        CALL LECFP
        CMP  AL,'-'
        JZ   SGNDEC16IN1
        CMP  AL,'+'
        JZ   SGNDEC16IN2
        JMP  SGNDEC16IN3
SGNDEC16IN1:    MOV  CH,0FFH
SGNDEC16IN2:    CALL LECFP
SGNDEC16IN3:    SUB  AL,30H
        JL   SGNDEC16IN4
        CMP  AL,9
        JG   SGNDEC16IN4
        CBW
        PUSH CX
        PUSH AX
        MOV  AX,DX
        MOV  CX,10
        MUL  CX
        MOV  DX,AX
        POP  AX
        ADD  DX,AX
        POP  CX
        JMP  SGNDEC16IN2
SGNDEC16IN4:    CMP  CH,0
        JE   SGNDEC16IN5
        NEG  DX
SGNDEC16IN5:    RET
;                       FPINDIGIT
FPINDIGIT: PUSH  DI
    PUSH AX
    MOV  AL,0
    MOV  CX,15
FPINDIGIT1:
        MOV  [DI],AL
    INC  DI
    LOOP FPINDIGIT1
    POP  AX
    POP  DI
    MOV  9[DI],AL
    RET
;           MULTIPLICATION PAR 10
FPTMUL:     MOV  CX,5
    MOV  DX,0
FPTMUL1: PUSH   CX
    MOV  AX,DX
    XCHG AX,[DI]
    MOV  CX,10
    MUL  CX
    ADD  [DI],AX
    ADD  DI,2
    POP  CX
    LOOP FPTMUL1
    RET
;           DIVISION PAR 10
FPTDIV:     MOV  CX,4
FPTDIV1: SAL    WORD PTR [DI],1
    RCL  WORD PTR 2[DI],1
    RCL  WORD PTR 4[DI],1
    RCL  WORD PTR 6[DI],1

    RCL  WORD PTR 8[DI],1
    DEC  WORD PTR 11[DI]
    LOOP FPTDIV1
    MOV  CX,5
    MOV  DX,0
    ADD  DI,8
FPTDIV2: PUSH   CX
    MOV  AX,[DI]
    MOV  CX,10
    DIV  CX
    MOV  [DI],AX
    SUB  DI,2
    POP  CX
    LOOP FPTDIV2
    RET
;               NORMALISATION DU FP
FPTNORM:    CMP  WORD PTR 08[DI],80H
        JL   FPTNORM1
        SAR  WORD PTR 08[DI],1
        RCR  WORD PTR 06[DI],1
        RCR  WORD PTR 04[DI],1
        RCR  WORD PTR 02[DI],1
        RCR  WORD PTR 00[DI],1
        INC  WORD PTR 11[DI]
        JMP  FPTNORM
FPTNORM1:   RET
;
;
;       NORMALISATION APRES OPERATION
FLTNORM:    MOV  AL,BYTE PTR 4[SI]
        SAL  AL,1
        JNC  FLTNORM1
        SHR  BYTE PTR 04[SI],1
        RCR  WORD PTR 02[SI],1
        RCR  WORD PTR 00[SI],1
        INC  BYTE PTR 05[SI]
        JMP  FLTNORM
FLTNORM1:   MOV  AL,BYTE PTR 4[SI]
        CMP  AL,40H
        JGE  FLTNORM2
        SAL  WORD PTR [SI],1
        RCL  WORD PTR 2[SI],1
        RCL  BYTE PTR 4[SI],1
        DEC  BYTE PTR 5[SI]
```

```
        JMP  FLTNORM1
FLTNORM2: RET
;               CONVERSION EN INTERNE
EXTINT:
     MOV  AX,WORD PTR FPTEMP1
     OR   AX,AX
     JZ   EXTINT5
     STC
EXTINT5:
     MOV  AX,WORD PTR FPTEMP1+2
     RCL  AX,1
     MOV  AX,WORD PTR FPTEMP1+4
     ADC  AX,0
     MOV  WORD PTR NUMB+0,AX
     MOV  DX,AX
     MOV  AX,WORD PTR FPTEMP1+6
     ADC  AX,0
     OR   DX,AX
     MOV  WORD PTR NUMB+2,AX
     MOV  AX,WORD PTR FPTEMP1+8
     OR   DX,AX
     AND  AX,007FH
     MOV  WORD PTR NUMB+4,AX
     MOV  AL,BYTE PTR FPTEMP1+10
     AND  AL,80H
     OR   BYTE PTR NUMB+4,AL
;
     MOV  AX,WORD PTR FPTEMP1+11
     CMP  AX,-128
     JL   EXTINT1
     CMP  AX,127
     JG   EXTINT2
     ADD  AX,80H
     CMP  DX,0
     JNE  EXTINT3
     MOV  AL,0
EXTINT3: MOV   BYTE PTR NUMB+5,AL
     JMP  EXTINT4
EXTINT1: MOV   BYTE PTR FPERR,10H
     CALL      ERRFP
     JMP EXTINT4
EXTINT2: MOV   BYTE PTR FPERR,20H
     CALL      ERRFP
EXTINT4: RET
;
;
;                   INTEXT
INTEXT:   MOV  WORD PTR FPTEMP1,0
        MOV    WORD PTR FPTEMP1+2,0
     MOV  AX,WORD PTR NUMB+0
     MOV  WORD PTR FPTEMP1+4,AX
     MOV  AX,WORD PTR NUMB+2
     MOV  WORD PTR FPTEMP1+6,AX
     MOV  AX,WORD PTR NUMB+4
     AND  AX,007FH
     OR   AX,0
     MOV  WORD PTR FPTEMP1+8,AX
     MOV  AL,BYTE PTR NUMB+4
     AND  AL,80H
     MOV  BYTE PTR FPTEMP1+10,AL
     MOV  AL,BYTE PTR NUMB+5
     MOV  AH,0
     SUB  AX,80H
     MOV  WORD PTR FPTEMP1+11,AX
     RET
;;
;                   SAISIE FLOTTANT
                    ---------------
;
FPIN:     PUSH DI
     PUSH SI
     PUSH DX
     PUSH CX
     PUSH AX
     LEA  DI,FPTEMP1
     MOV  AL,0
     CALL FPINDIGIT
;
     MOV  DECFLAG,0
     MOV  DECEXP,0
     CALL LECFP
     CMP  AL,'-'
     JZ   FPIN1
     CMP  AL,'+'
     JZ   FPIN2
     JMP  FPIN3
;
FPIN1: MOV  BYTE PTR FPTEMP1+10,80H
FPIN2: CALL LECFP
FPIN3: CMP  AL,'.'
     JNE  FPIN4
;
     CMP  DECFLAG,0
     JNE  FPIN5
     MOV  DECFLAG,0FFH
     JMP  FPIN2
FPIN4: SUB  AL,30H
     JL   FPIN5
     CMP  AL,9
     JG   FPIN5
     JMP  FPIN6
;
FPIN5: JMP  FPIN15
FPIN6: LEA  DI,FPTEMP2
     CALL FPINDIGIT
;
     LEA  DI,FPTEMP1
     CALL FPTMUL
     MOV  CX,WORD PTR FPTEMP1+11
     SUB  CX,WORD PTR FPTEMP2+11
     JE   FPIN11
     JG   FPIN9
FPIN7: NEG  CX
FPIN8:        SAR     WORD PTR FPTEMP1+08,1
     RCR  WORD PTR FPTEMP1+06,1
          RCR WORD PTR FPTEMP1+04,1
     RCR  WORD PTR FPTEMP1+2,1
     RCR  WORD PTR FPTEMP1+0,1
     LOOP FPIN8
;
     MOV  AX,WORD PTR FPTEMP2+11
     MOV  WORD PTR FPTEMP1+11,AX
     JMP  FPIN11
FPIN9:
;
FPIN10:      SAR  WORD PTR FPTEMP2+08,1
     RCR  WORD PTR FPTEMP2+06,1
             RCR  WORD PTR FPTEMP2+04,1
     RCR  WORD PTR FPTEMP2+2,1
     RCR  WORD PTR FPTEMP2+0,1
     LOOP FPIN10
;
     MOV  AX,WORD PTR FPTEMP1+11
     MOV  WORD PTR FPTEMP2+11,AX
     JMP  FPIN11
FPIN11:     MOV  CX,5
     LEA  DI,FPTEMP1
     LEA  SI,FPTEMP2
     CLC
FPIN12:     MOV  AX,[SI]
     INC  SI
     INC  SI
     ADC  [DI],AX
     INC  DI
     INC  DI
     LOOP FPIN12
;
     LEA  DI,FPTEMP1
     CALL FPTNORM
;
FPIN13:     CMP  DECFLAG,0
     JE   FPIN14
     DEC  DECEXP
FPIN14:     JMP  FPIN2
FPIN15:     ADD  AL,30H
     AND  AL,5FH
     CMP  AL,'E'
     JNE  FPIN16
     CALL SGNDEC16IN
     ADD  WORD PTR DECEXP,DX
FPIN16:     MOV  CX,DECEXP

     CMP  CX,0
     JG   FPIN17
     JL   FPIN18
     JMP  FPIN20
FPIN17:     PUSH CX
```

```
        LEA  DI,FPTEMP1
        CALL FPTMUL
        LEA  DI,FPTEMP1
        CALL FPTNORM
        POP  CX
        LOOP FPIN17
        JMP  FPIN20
FPIN18:      NEG CX
FPIN19:      PUSH CX
        LEA  DI,FPTEMP1
        CALL FPTDIV
        LEA  DI,FPTEMP1
        CALL FPTNORM
        POP  CX
        LOOP FPIN19
FPIN20:      CALL EXTINT
             POP  AX
        POP  CX
        POP  DX
        POP  SI
        POP  DI
        RET
;
;
;
;            EFFACEMENT DU BUFFER DE SORTIE
EFFSORT:     PUSH DI
             PUSH CX
             PUSH AX
             LEA  DI,BUFCAROUT
             MOV  CX,20
             MOV  AX,0
EFFSORT1:    MOV  WORD PTR [DI],AX
             INC  DI
             INC  DI
             LOOP EFFSORT1
             POP  AX
             POP  CX
             POP  DI
             RET
STOOUT:           PUSH DI
             PUSH CX
             LEA  DI,BUFCAROUT+1
             MOV  CL,BYTE PTR BUFCAROUT
             CBW
             ADD  DI,CX
             MOV  BYTE PTR [DI],AL
             INC  BYTE PTR BUFCAROUT
             POP  CX
             POP  DI
             RET
;                 AFFICHAGE FLOTTANT
TDECSHOW:    CMP  DECSIGN,0
             MOV  AL,' '
             JE   TDECSHOW1
             MOV  AL,'-'
TDECSHOW1:        CALL STOOUT
TDECSHOW2:        LEA  SI,DECBUFF+21
             MOV  AL,[SI]
             DEC  SI
             ADD  AL,30H
             CALL STOOUT
             MOV  AL,'.'
             CALL STOOUT
             MOV  CX,00H   ; NB DE DIGITS DECIMAUX
TDECSHOW3:        MOV  AL,[SI]
             DEC  SI
             ADD  AL,30H
             CALL STOOUT
             LOOP TDECSHOW3
;
             MOV  AL,'E'
             CALL STOOUT
             MOV  DX,DECEXP
             CMP  DX,0
             MOV  AL,'+'
             JGE  TDECSHOW4
             NEG  DX
             MOV  AL,'-'
TDECSHOW4:        CALL STOOUT
             CALL DEC16OUT
             RET
;
;                        DEC16OUT
;
DEC16OUT:    PUSH DI
             PUSH DX
             PUSH CX
             PUSH AX
             MOV  CX,0
             LEA  DI,TBUFF
DEC16OUT1:   PUSH CX
             MOV  AX,DX
             MOV  DX,0
             MOV  CX,10
             DIV  CX
             XCHG AX,DX
             ADD  AL,30H
             MOV  [DI],AL
             INC  DI
             POP  CX
             INC  CX
             CMP  DX,0
             JNZ  DEC16OUT1
DEC16OUT2:   DEC  DI
             MOV  AL,BYTE PTR [DI]
             CALL STOOUT
             LOOP DEC16OUT2
             POP  AX
             POP  CX
             POP  DX
             POP  DI
             RET
; CONVERTION 48 BITS EN BCD
; -------------------------
BIN802DEC:   PUSH DI
             MOV  AL,0
             MOV  CX,25
BIN802DEC1:  MOV  [DI],AL
             INC  DI
             LOOP BIN802DEC1
             POP  DI
BIN802DEC2:  PUSH SI
             MOV  BX,0
             MOV  CX,5
             MOV  DX,0
             ADD  SI,8
BIN802DEC3:  PUSH CX
             MOV  AX,[SI]
             MOV  CX,10
             DIV  CX
             MOV  [SI],AX
             OR   BX,AX
             SUB  SI,2
             POP  CX
             LOOP BIN802DEC3
             MOV  [DI],DL
             INC  DI
             POP  SI
             CMP  BX,0
             JNZ  BIN802DEC2
             RET
;            NORMALISATION DECIMALE
;            ----------------------
DECNORM:     CMP  BYTE PTR 22[DI],0
             JE   DECNORM2
             MOV  AL,[DI]
             ADD  AL,AL
             MOV  AH,0
             AAA
             MOV  CX,24
DECNORM1:    MOV  AL,1[DI]
             ADC  AL,AH
             MOV  AH,0
             AAA
             MOV  [DI],AL
             INC  DI
             LOOP DECNORM1
             INC  DECEXP
DECNORM2:    RET
;
;            MOITIE DECIMALE
;            ---------------
DECHALF:     MOV  CX,25
             MOV  AL,0
DECHALF1:    XCHG AL,[DI]
```

```
        INC  DI
        LOOP DECHALF1
        DEC  DECEXP
        MOV  CX,25
        MOV  AH,0
DECHALF2: PUSH CX
        DEC  DI
        MOV  AL,[DI]
        MOV  CL,2
        AAD
        DIV  CL
        MOV  [DI],AL
        POP  CX
        LOOP DECHALF2
        RET
;
;       DOUBLE DECIMAL
;       --------------
DECDOUBLE:      MOV  CX,25

        MOV  AH,0
DECDOUBLE1:     MOV  AL,[DI]
        SAL  AL,1
        ADD  AL,AH
        AAM
        MOV  [DI],AL
        INC  DI
        LOOP DECDOUBLE1
        RET
;
;       SORTIE FLOTTANT
;       ---------------
FPOUT:          PUSH DI
        PUSH SI
        PUSH DX
        PUSH BX
        PUSH CX
        PUSH AX
        MOV  AX,WORD PTR NUMB
        OR   AX,WORD PTR NUMB+2
        OR   AX,WORD PTR NUMB+4
        JNZ  FPOUT1
        MOV  AL,'0'
        CALL STDOUT
        JMP  FPOUT6
FPOUT1:         CALL    INTEXT
        MOV  DECEXP,21

        MOV  AL,BYTE PTR FPTEMP1+10
        MOV  DECSIGN,AL
        LEA  SI,FPTEMP1
        LEA  DI,DECBUFF
        CALL BIN802DEC
        MOV  CX,WORD PTR FPTEMP1+11
        SUB  CX,72
        CMP  CX,0
        JL   FPOUT2
        JG   FPOUT4
        JMP  FPOUT5

FPOUT2:         NEG  CX
FPOUT3:         PUSH CX
        LEA  DI,DECBUFF
        CALL DECHALF
        LEA  DI,DECBUFF
        CALL DECNORM
        POP  CX
        LOOP FPOUT3
        JMP  FPOUT5
FPOUT4:         PUSH CX
        LEA  DI,DECBUFF
        CALL DECDOUBLE
        LEA  DI,DECBUFF
        CALL DECNORM
        POP  CX
        LOOP FPOUT4
        JMP  FPOUT5
FPOUT5:         CALL TDECSHOW
FPOUT6:         POP  AX
        POP  CX
        POP  BX
        POP  DX
        POP  SI
        POP  DI
        RET

;
;
; ADDITION
; SI = OPER1
; DI = OPER2
; LE RESULTAT EST DANS OPER1
; LES SIGNES SONT TESTES AVANT L'OPERATION
;
MBINADD:
  PUSH SI
  PUSH DI

  MOV  CX,2
  CLC
MBINADD1:
  MOV  AX,[SI]
  ADC  AX,[DI]
  INC  DI
  INC  DI
  MOV  [SI],AX
  INC  SI
  INC  SI
  LOOP MBINADD1
  MOV  AL,[SI]
  ADC  AL,[DI]
  MOV  [SI],AL
  POP  DI
  POP  SI
  RET
;
; SOUSTRACTION
;
MBINSUB:
  PUSH SI
  PUSH DI
  MOV  CX,2
  CLC
MBINSUB1:
  MOV  AX,[SI]
  SBB  AX,[DI]
  INC  DI
  INC  DI
  MOV  [SI],AX
  INC  SI
  INC  SI
  LOOP MBINSUB1
  MOV  AL,[SI]
  SBB  AL,[DI]
  MOV  [SI],AL
  POP  DI
  POP  SI
  RET
;
; MULTIPLICATION
;
MBINMUL:
  PUSH SI
  PUSH DI
  PUSH BX
  PUSH CX
  PUSH AX
  LEA  BX,RESMUL
  LEA  SI,OPER1
  LEA  DI,OPER2
  PUSH BX
  MOV  AX,0
  MOV  CX,06
  CLD
MBINMUL1:
  MOV  [BX],AX
  INC  BX
  INC  BX
  LOOP MBINMUL1
  POP  BX
  MOV  CX,3
MBINMUL2:
  PUSH CX
  MOV  DX,[SI]
  INC  SI
  INC  SI
```

```
     PUSH BX
     PUSH DI
     MOV  CX,3
MBINMUL3:
     PUSH CX
     PUSH DX
     MOV  AX,[DI]
     INC  DI
     INC  DI
     MUL  DX
     ADD  [BX],AX
     INC  BX
     INC  BX
     ADC  [BX],DX
     POP  DX
     POP  CX
     LOOP MBINMUL3
     POP  DI
     POP  BX
     INC  BX
     INC  BX
     POP  CX
     LOOP MBINMUL2
     POP  AX
     POP  CX
     POP  BX
     POP  DI
     POP  SI
     RET
;
;
;         DIVISION
;
;
;    SHIFT LE DIVIDENDE 1 POSITION A GAUCHE
;    SI CONTIENT LE DIVISEUR
;    DI CONTIENT LE DIVIDENDE
;    BX CONTIENT LE QUOTIENT
;
DIVSAL:
     PUSH DI
     PUSH CX
     MOV  CX,3
     CLC
DIVSAL1:
     RCL  WORD PTR  [DI],1
     INC  DI
     INC  DI
     LOOP DIVSAL1
     POP  CX
     POP  DI
     RET
;
;
;
;
DIVSUB:
     PUSH SI
     PUSH DI
     PUSH BX
     PUSH CX
     LEA  BX,TEMPDIV
     CLC
     MOV  CX,3
DIVSUB1:
     MOV  DX,[SI]
     INC  SI
     INC  SI
     MOV  AX,[DI]
     INC  DI
     INC  DI
     SBB  AX,DX
     MOV  [BX],AX
     INC  BX
     INC  BX
     LOOP DIVSUB1
     POP  CX
     POP  BX
     POP  DI
     POP  SI
     RET
;
;    CHGEMENT DE DIVIDENDE
MOVEIT:        PUSH DI
       PUSH SI
       PUSH CX
       MOV  CX,3
       LEA  SI,TEMPDIV
       CLD
       REP  MOVSW
       POP  CX
       POP  SI
       POP  DI
       RET
;
;
QUOTSHL:
  PUSH BX
  PUSH CX
  MOV  CX,3
QUOTSHL1:
  RCL  WORD PTR  [BX],1
  INC  BX
  INC  BX
  LOOP QUOTSHL1
  POP  CX
  POP  BX
  RET
;
;
; DIVISION
;
MBINDIV:
  PUSH SI
  PUSH DI
  PUSH BX
  PUSH CX
  PUSH AX
  MOV  CX,39
MBIND1:    PUSH CX
  CALL DIVSUB
  JC   MBIND2
  CALL MOVEIT
  STC
  JMP  MBIND3
MBIND2:    CLC
MBIND3:    CALL QUOTSHL
  CLC
  CALL      DIVSAL
  POP  CX
  LOOP MBIND1
  POP  AX
  POP  CX
  POP  BX
  POP  DI
  POP  SI
  RET
;
;
;           AJUSTEXP
; SI = OPER1
; DI = OPER2
; AL ET AH CONTIENNENT LES EXPOSANTS DE OPER1 ET OPER2
;
AJUSTEXP:
       SUB  AL,AH
       JZ   AJUSTEXP1
       CMP  AL,0
       JL   AJUSTEXP2
       CMP  AL,39
       JG   AJUSTEXP3
       MOV  CL,AL
       SUB  CH,CH
AJUST1:         SAR  BYTE PTR 4[DI],1
       RCR  WORD PTR 2[DI],1
       RCR  WORD PTR 0[DI],1
       INC  BYTE PTR 5[DI]
       LOOP AJUST1
       JMP  AJUSTEXP1
AJUSTEXP2:  MOV  CL,AL
       NEG  CX
       SUB  CH,CH
       CMP  CX,39
       JG   AJUSTEXP3
AJUST2:         SAR  BYTE PTR 4[SI],1
       RCR  WORD PTR 2[SI],1
```

```
        RCR  WORD PTR 0[SI],1
        INC  BYTE PTR 5[SI]
        LOOP AJUST2
        JMP  AJUSTEXP1
AJUSTEXP3:      MOV  AX,2
        JMP  AJUSTEXP4
AJUSTEXP1:      MOV  AX,0
AJUSTEXP4:      RET
;
;
;    AVANT SI = OPER1 DI = OPER2
;    APRES DI = OPER1 SI = OPER2
;                   FXCH6
FXCH6:          PUSH DI
        PUSH SI
        CLD
                MOV  CX,3
FXCH61:         MOV  AX,[SI]
        XCH6 AX,[DI]
        MOV  [SI],AX
        ADD  SI,2
        ADD  DI,2
        LOOP FXCH61
        POP  SI
        POP  DI
        RET
;
;                          FADD
;
;
FADD:           LEA  SI,OPER1
        LEA  DI,OPER2
        MOV  AX,WORD PTR 4[SI]
        OR   AX,AX
        JNZ  CONTADD
        PUSH SI
        PUSH DI
        MOV  CX,3
LPADD:          MOV  AX,[DI]
        MOV  [SI],AX
        ADD  DI,2
        ADD  SI,2
        LOOP LPADD
        POP  DI
        POP  SI
        JMP  FADD6
CONTADD: MOV     AL,BYTE PTR 4[SI]
        AND  BYTE PTR 4[SI],7FH
        AND  AL,80H
        MOV  SGNOPER1,AL
        MOV  AL,BYTE PTR 4[DI]
        AND  BYTE PTR 4[DI],7FH
        AND  AL,80H
        MOV  SGNOPER2,AL
        MOV  AL,BYTE PTR 5[SI]
        MOV  AH,BYTE PTR 5[DI]
        CALL AJUSTEXP
        OR   AX,AX
        JNZ  FADD6
        MOV  AL,SGNOPER1
        MOV  AH,SGNOPER2
        SUB  AL,AH
        JZ   FADD3
        CALL FCMP
        OR   AX,AX
        JNZ  FADD4
        CALL FXCHG
        MOV  AL,SGNOPER1
        MOV  AH,SGNOPER2
        MOV  SGNOPER1,AH
        MOV  SGNOPER2,AL
        JMP  FADD7
FADD3:          CALL MBINADD
        JMP  FADD5
FADD4:          CMP  AX,80H
        JZ   ZEROSUB
FADD7:          CALL MBINSUB
FADD5:          MOV  AX,WORD PTR 4[SI]
        OR   AX,AX
        JZ   FADD6
        CALL FLTNORM
        MOV  AL,SGNOPER1
        OR   BYTE PTR 4[SI],AL
```

```
FADD6:          RET
ZEROSUB:        PUSH SI
        MOV  AX,0
        LEA  SI,OPER1
        MOV  CX,3
BLZERO:              MOV  [SI],AX
        ADD  SI,2
        LOOP BLZERO
        POP  SI
        JMP  FADD6
;
;
;
;       SI = OPER1
;       DI = OPER2
; SI > DI ==> AX=1
; DI > SI ==> AX=0
; DI = SI ==> AX=80H
FCMP:           PUSH      SI
        PUSH DI
        MOV  AL,BYTE PTR 4[SI]
        MOV  AH,BYTE PTR 4[DI]
        CMP  AL,AH
        JG   FCMP1
        JL   FCMP2
        MOV  CX,2
        ADD  SI,2
        ADD  DI,2
FCMPB:          MOV  AX,WORD PTR [SI]
        MOV  DX,WORD PTR [DI]
        SUB  SI,2
        SUB  DI,2
        CLC
        RCR  AX,1
        CLC
        RCR  DX,1
        SUB  AX,DX
        JG   FCMP1
        JL   FCMP2
        DEC  CX
        JNZ  FCMPB
        MOV  AX,80H
        JMP  FCMP5
FCMP1:          MOV  AX,1
        JMP  FCMP5
FCMP2:          MOV  AX,0
FCMP5:          POP  DI
        POP  SI
        RET
;
;
;
;
;               MULTIPLICATION
FMULT:          MOV  AX,WORD PTR OPER1+4
        OR   AX,0
        JZ   ZMULT
        MOV  MULEXP1,AH
        AND  AL,80H
        MOV  SGNOPER1,AL
        AND  WORD PTR OPER1+4,007FH
        MOV  AX,WORD PTR OPER2+4
        OR   AX,0
        JNZ  SUITMUL
ZMULT:          PUSH DI
        MOV  AX,0
        LEA  DI,OPER1
        MOV  CX,3
MULZ:           MOV  WORD PTR [DI],AX
        INC  DI
        INC  DI
        LOOP MULZ
        POP  DI
        JMP  FINMUL
SUITMUL:        MOV  MULEXP2,AH
        AND  AL,80H
        MOV  SGNOPER2,AL
        AND  WORD PTR OPER2+4,007FH
        CALL MBINMUL
        PUSH DI
        LEA  DI,RESMUL
        CALL MULNORM
        POP  DI
```

```
        MOV  CX,5
        CLD
        PUSH SI
        PUSH DI
        LEA  SI,RESMUL+5
        LEA  DI,OPER1
    REP MOVSB
        POP  DI
        POP  SI
        MOV  AL,MULEXP1
        MOV  AH,MULEXP2
        SUB  AL,80H
        ADD  AL,AH
        MOV  BYTE PTR OPER1+5,AL
        MOV  AL,SGNOPER1
        MOV  AH,SGNOPER2
        XOR  AL,AH
        OR   BYTE PTR OPER1+4,AL
FINMUL:      RET
;
;
;
;   NORMALISATION DE LA MULTIPLICATION
MULNORM: MOV  AL,BYTE PTR 09[DI]
         SAL  AL,1
         JNC  MULNORM1
         SAR  WORD PTR 08[DI],1
         RCR  WORD PTR 06[DI],1
         RCR  WORD PTR 04[DI],1
         RCR  WORD PTR 02[DI],1
         RCR  WORD PTR 00[DI],1
         INC  MULEXP1
         JMP     MULNORM
MULNORM1: CMP BYTE PTR 09[DI],40H
         JGE  MULNORM2
         SAL  WORD PTR 00[DI],1
         RCL  WORD PTR 02[DI],1
         RCL  WORD PTR 04[DI],1
         RCL  WORD PTR 06[DI],1
         RCL  WORD PTR 08[DI],1
         DEC  MULEXP1
         JMP  MULNORM1
MULNORM2: RET
;
;
;        ZERO DU QUOTIENT
NULQ:    PUSH BX
     PUSH AX
     MOV  AX,0
     MOV  CX,3
NULQ1:   MOV  [BX],AX
     INC  BX
     INC  BX
     LOOP NULQ1
     POP  AX
     POP  BX
     RET
;                FDIVISION
;
FDIV:        PUSH SI
         PUSH DI
         PUSH BX
         LEA  SI,OPER2
         LEA  DI,OPER1
         LEA  BX,QUOTIENT
         CALL NULQ
             MOV  AX,WORD PTR OPER1+4
         MOV  MULEXP1,AH
         AND  AL,80H
         MOV  SGNOPER1,AL
         AND  WORD PTR OPER1+4,007FH
         MOV  AX,WORD PTR OPER2+4
         MOV  MULEXP2,AH
         AND  AL,80H
         MOV  SGNOPER2,AL
         AND  WORD PTR OPER2+4,007FH
         CALL MBINDIV
         CALL DIVNORM
         MOV  AL,MULEXP1
         MOV  AH,MULEXP2
         SUB  AL,AH
         ADD  AL,82H
         MOV  BYTE PTR QUOTIENT+5,AL
         MOV  AL,SGNOPER1
         MOV  AH,SGNOPER2
         XOR  AL,AH
         OR   BYTE PTR QUOTIENT+4,AL
         POP  BX
         POP  DI
         POP  SI
         RET
;
;            DIVNORM
DIVNORM:     MOV  AL,BYTE PTR QUOTIENT+4
         SAL  AL,1
         JNC  DIVNORM1
         SAR  WORD PTR QUOTIENT+4,1
         RCR  WORD PTR QUOTIENT+2,1
         RCR  WORD PTR QUOTIENT,1
         INC  MULEXP1
         JMP     DIVNORM
DIVNORM1:    CMP  WORD PTR QUOTIENT+4,40H
         JGE  DIVNORM2
         SAL  WORD PTR QUOTIENT,1
         RCL  WORD PTR QUOTIENT+2,1
         RCL  WORD PTR QUOTIENT+4,1
         DEC  MULEXP1
         JMP  DIVNORM1
DIVNORM2:    RET
;
;
;
;
;        DIVISION PAR 0 ?
CMP0:        PUSH DI
         PUSH CX
         LEA  DI,OPER2
         MOV  CX,3
         MOV  AX,0
CMP01:       CMP  AX,WORD PTR [DI]
         JNE  CMP02
         INC  DI
         INC  DI
         LOOP CMP01
         MOV  FPERR,40H
         JMP  CMP03
CMP02:       MOV  FPERR,0
CMP03:       POP  CX
         POP  DI
         RET
;
;        ASC>FP
   DB    86H
   DB    'ASC>F'
   DB    'P'+80H
   DW    FPINP-7
ASCTFP DW    $+2
   POP  CX
   POP  AX
   PUSH SI
   PUSH DI
   LEA  DI,BUFCARIN+2
   MOV  SI,AX
   MOV  CH,0
   MOV  AX,DS
   MOV  ES,AX
   MOV  WORD PTR BUFCARIN,CX
   REPNZ     MOVSB
   POP  DI
   POP  SI
   CALL FPIN
   PUSH WORD PTR NUMB
   PUSH WORD PTR NUMB+2

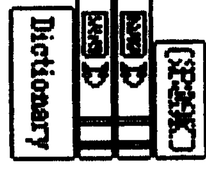
   PUSH WORD PTR NUMB+4
   NEXT
;
;        FP>ASC
;
   DB    86H
   DB    'FP>AS'
   DB    'C'+80H
   DW    ASCTFP-9
FPTASC DW    $+2
   POP  WORD PTR NUMB+4
   POP  WORD PTR NUMB+2
```

```
        POP  WORD PTR NUMB+0
        CALL EFFSORT
        CALL FPOUT
        LEA  AX,BUFCAROUT
        PUSH AX
        NEXT
;
;               F+
        DB   82H
        DB   'F'
        DB   '+'+80H
        DW   FPTASC-9
FPLUS       DW   $+2
SBA: POP   AX
        POP  WORD PTR OPER2+2
        POP  WORD PTR OPER2+0
        POP  WORD PTR OPER1+4
        POP  WORD PTR OPER1+2
        POP  WORD PTR OPER1+0
        OR   AX,AX
        JZ   FSBA
        MOV  WORD PTR OPER2+4,AX
        PUSH SI
        PUSH DI
        CALL FADD
        POP  DI
        POP  SI
FSBA:        PUSH WORD PTR OPER1+0
        PUSH WORD PTR OPER1+2
        PUSH WORD PTR OPER1+4
        NEXT
;
;
;               F-
        DB   82H
        DB   'F'
        DB   '-'+80H
        DW   FPLUS-5
FSUB DW      $+2
        POP  AX
        OR   AX,0
        JZ   FINSUB
        XOR  AX,80H
FINSUB:      PUSH AX
             JMP  SBA
;
;
;            MULTIPLICATION
        DB   82H
        DB   'F'
        DB   '*'+80H
        DW   FSUB-5
FSTAR        DW   $+2
        POP  AX
        POP  WORD PTR OPER1+2
        POP  WORD PTR OPER1+0
        POP  BX
        POP  WORD PTR OPER2+2
        POP  WORD PTR OPER2+0
        OR   AX,AX
        JZ   MULZERO
        OR   BX,BX
        JZ   MULZERO
        MOV  WORD PTR OPER1+4,AX
        MOV  WORD PTR OPER2+4,BX
        PUSH SI
        PUSH DI
        CALL FMULT
        POP  DI
        POP  SI
        PUSH WORD PTR OPER1+0
        PUSH WORD PTR OPER1+2
        PUSH WORD PTR OPER1+4
        NEXT
MULZERO:     MOV  AX,0
             PUSH AX
             PUSH AX
             PUSH AX
        NEXT
;
;
;            DIVISION
;
```

```
        DB   82H
        DB   'F'
        DB   '/'+80H
        DW   FSTAR-5
FSLAS   DW   $+2
        POP  AX
        POP  WORD PTR OPER2+2
        POP  WORD PTR OPER2+0
        POP  BX
        POP  WORD PTR OPER1+2
        POP  WORD PTR OPER1+0
        OR   AX,AX
        JZ   MULZERO
        OR   BX,BX
        JZ   MULZERO
        MOV  WORD PTR OPER1+4,BX
        MOV  WORD PTR OPER2+4,AX
        CALL FDIV
        PUSH WORD PTR QUOTIENT+0
        PUSH WORD PTR QUOTIENT+2
        PUSH WORD PTR QUOTIENT+4
        NEXT
;
;
;            FDUP
;
        DB   84H
        DB   'FDU'
        DB   'P'+80H
        DW   FSLAS-5
FDUPL   DW   $+2
        POP  AX
        POP  DX
        POP  BX
        PUSH BX
        PUSH DX
        PUSH AX
        PUSH BX
        PUSH DX
        PUSH AX
        NEXT
;
;
;            F0
        DB   82H
        DB   'F'
        DB   '0'+80H
        DW   FDUPL-7
FZER    DW   $+2
        MOV  DX,0
        PUSH DX
        PUSH DX
        PUSH DX
        NEXT
;;
;
;            INT
        DB   83H
        DB   'IN'
        DB   'T'+80H
        DW   FZER-5
INTE    DW   $+2
        POP  BX
        POP  AX
        POP  DX
        PUSH BX
        AND  BL,80H
        MOV  SGNOPER1,BL
        POP  BX
        MOV  DX,BX
        MOV  CL,DH
        AND  DX,007FH
        MOV  CH,0
        SUB  CX,88H
        CMP  CX,0
        JL   INT1
        JG   INT3
        JE   INT4
INT1:   NEG  CX
INT2:   SAR  DX,1
        RCR  AX,1
        LOOP INT2
        JMP  INT4
```

```
INT3:     SAL  AX,1
     RCL  DX,1
     LOOP INT3
INT4:     MOV  AL,SGNOPER1
     CMP  AL,80H
     JNE  INT5
     NEG  DX
INT5:     PUSH DX
     NEXT
;
;
;         FLOAT
     DB   85H
     DB   'FLOA'
     DB   'T'+80H
     DW   INTE-6
FLOAT     DW   $+2
     POP  DX
     MOV  BX,DX
     MOV  AX,0
     CMP  DX,0
     JZ   FLOAT4
     JG   FLOAT1
     NEG  DX
FLOAT1:   MOV  CX,9800H
FLOAT2:   CMP  AX,0040H
     JGE  FLOAT3
     SAL  DX,1
     RCL  AX,1
     DEC  CH
     JMP  FLOAT2
FLOAT3:   AND  AX,007FH
     OR   AX,CX
     CMP  BX,0
     JG   FLOAT4
     OR   AL,80H
FLOAT4:   MOV  BX,0
     PUSH BX
     PUSH DX
     PUSH AX
     NEXT
;
;

     DB   85H
     DB   'FOVE'
     DB   'R'+80H
     DW   FLOAT-8
FOVER     DW   $+2
     MOV  AX,SS
     MOV  DS,AX
     MOV  ES,AX
     MOV  DX,SI
     MOV  DI,SP
     MOV  SI,DI
     ADD  SI,10
     SUB  DI,2
     STD
     MOV  CX,3
     REP  MOVSW
     CLD
     SUB  SP,6
     MOV  SI,DX
     MOV  AX,CS
     MOV  DS,AX
     NEXT
;
;
     DB   82H
     DB   'F'
     DB   '@'+80H
     DW   FOVER-8
FAT  DW   $+2
     POP  BX
     ADD  BX,4
     MOV  CX,3
FAT1:     PUSH [BX]
     SUB  BX,2
     LOOP FAT1
     NEXT
;
;
     DB   82H
     DB   'F'
     DB   '!'+80H
     DW   FAT-5
FST  DW   $+2
     POP  BX
     MOV  CX,3
FST1: POP  [BX]
     ADD  BX,2
     LOOP FST1
     NEXT
;
;
     DB   85H
     DB   'FSWA'
     DB   'P'+80H
     DW   FST-5
FSWAP DW   $+2
     POP  AX
     POP  BX
     POP  CX
     POP  DX
     POP  WORD PTR OPER1
     POP  WORD PTR OPER2
     PUSH CX
     PUSH BX
     PUSH AX
     PUSH WORD PTR OPER2
     PUSH WORD PTR OPER1
     PUSH DX
     NEXT
;
```



COURRIER: Cher Secrétaire,

Je viens de recevoir le dernier numéro de JEDI et c'est avec une grande tristesse que je verrai disparaître JEDI qui m'apporte des informations intéressantes. J'espère qu'il n'en sera rien et que nous trouverons à fournir de quoi publier. Je pourrai éventuellement vous faire parvenir un résumé du type de travail que nous avons développé en FORTH. C'est assez particulier, mais je ne pourrai jamais joindre de listings comme vous le faites en général.

Je me suis abonné à JEDI car j'avais sélectionné le FORTH comme langage de développement. Nous avons travaillé avec MacFORTH sur MacINTOSH et maintenant nous envisageons de transcrire ce programme sur IBM/PC. Ce sera sans doute en FORTH LMI car il possède (d'après les documentations que j'ai pu recueillir) de nombreuses possibilités au niveau du graphisme (dessins de molécules), ensuite le NATIVE CODE OPTIMIZER me semble excellent. Est-ce qu'il serait possible de passer une annonce dans un (éventuel) futur numéro afin de savoir si un des adhérents l'utilise et si on obtient une rapidité d'exécution excellente?

J'aurai bien aimé trouver un article sur les rapidités des différents langages. Si vous le désirez, je pourrais écrire quelque chose sur mes "sentiments" par rapport au langage FORTH, ce que moi je trouve bien et moins bien, histoire de lancer une sorte de polémique? Je ne sais pas si ce serait intéressant?

Enfin, je peux vous donner des idées de mots FORTH que j'aimerais trouver dans JEDI, si des doués pourraient les écrire:

- GOTO (quand on a été nourri au BASIC, il est difficile de s'en passer. Un exemple avait été donné dans un numéro, mais pour le 6502).

- le passage d'arguments comme dans le MacFORTH. Les exemples donnés dans un précédent numéro sont différents. En MacFORTH on écrira par exemple:

```
VAR1 VAR2 VAR3 MOT
```

et mot a été défini ainsi:

```
: MOT  LOCALS: V3 V2 V1 ;
VAR3 - ) V3  VAR2 -) V2  VAR1 -) V1
```

ensuite quand on appelle V3, on a la valeur sur la pile.

Ce qui est génial en MacFORTH: il n'y a plus d'écran, on écrit des mots de la longueur qu'on veut. Le --) était plutôt lourd.

Quelqu'un pourrait-il écrire quelque chose sur les différents FORTH qui existent sur PC?

Bien cordialement
R.BARONE 13397 MARSEILLE CEDEX 13

REPONSE: Cher adhérent,

Pour l'instant, et grâce à votre courrier, JEDI survivra, mais il tient à tout le monde d'aider à maintenir la pression.

Concernant vos propositions d'idées d'articles, toute intervention provoquant une discussion est intéressante. Seul le débat sans contrainte est facteur de progrès, quelque soit l'opinion exprimée. Mais loin de nous l'idée de relancer un DROIT DE REPONSE écrit; c'est en provoquant la réflexion que l'on déclenche la créativité.

Concernant votre choix de la version FORTH LMI, je ne puis la critiquer. Le FORTH LMI n'est pas à la portée de toutes les bourses. Il continue à gérer des blocs, il dispose de fonctions graphiques mais n'est pas adapté aux cartes graphiques standard (IBM COLORCARD) du moins est-ce ce que j'ai constaté sur le PERSONA 1600 de LOGABAX. Si JEDI a choisi la version Laxen et Perry, c'est essentiellement parce que c'est le standard le plus utilisé, même s'il n'est pas aussi rapide que le FORTH LMI. La rapidité des langages reste un critère très subjectif. A moins de traiter des applications "temps réel", il faut toujours faire la part entre rapidité d'exécution et rapidité de développement. Ecrire une gestion de base de données est plus appropriée avec dBASE III. Pour ma part, FORTH est un outil venant compléter une panoplie logicielle en offrant le moyen de développer des modules qu'il me serait trop difficile à écrire en assembleur.

Une bonne nouvelle, je publierai bientôt dans JEDI les fonctions graphiques élémentaires en F83 pour tous systèmes sous MSDOS, ainsi que la gestion de variables locales tel que vous le suggérez:

```
  <DECLARE VAR TRUC   VAR MACHIN   LOCAL BIBULE
  <DEFINE ESSAI
     TRUC MACHIN * BIDULE LET  ( bidule:=truc*machin)
     >>      \ fin de définition
```

Concernant la suppressions des blocs, jetez un coup d'oeil sur l'article TURBO-FORTH. Il répond à vos souhaits et prouve une fois de plus que JEDI est I N D I S P E N S A B L E, mais que ce n'est pas toujours aux mêmes d'écrire des articles. La part de temps accordée au développement des programmes accompagnant les articles est souvent non négligeable, surtout quand on tient à proposer des sujets de valeur.

LE SECRETAIRE

COURRIER: J'espère pouvoir très bientôt vous faire parvenir un article qui sera consacré aux langages orientés objets et leur simulation en APL. La modestie nous empêche, plus souvent que l'envie, de partager nos petites découvertes. A bas la modestie!

DE SOZA
06300 NICE

COURRIER: Chers amis,

A la lecture de la lettre du Secrétaire qui était jointe au n°37, j'avais ressenti un certain malaise. La lettre du 21 septembre me confirme dans mes craintes. Il me semble que quelque chose ne tourne plus rond dans cette association à vocation très originale.

Malheureusement, il n'ets pas question pour moi de me rendre à la réunion du 26, pour la bonne raison que je reçois la convocation... le 26! Elle a été postée le 23, et timbrée au tarif "B", on ne peut guère en vouloir aux PTT qui ne sont pas en tort. La personne qui a expédié le courrier devait savoir ce qu'elle faisait!... Ou alors elle a fait preuve d'une insouciance un peu coupable.

Les décisions seront donc prises quand ma lettre sera lue par qui le voudra. Je le regrette. Et si j'avais eu envie de contribuer, malgré mon éloignement, à la tentative de sauvetage, je ne pourrai pas le faire.

Néanmoins, à tout hasard, je vous signale que toute association qui se veut solide doit tenir compte de la nature des associés. Ici, nous avons quelques fondus de trucs bizarres, le FORTH pour ma part. La dispersion sur le territoire national (Ndlr: et extra-territorial!) des adhérents est une conséquence inévitable de cette bizarrerie qui est notre lot commun. Et l'organisation interne de l'association devrait en tenir compte, en décentralisant les éléments de la structure qui peuvent ne pas en souffrir.

Il appert que tout le poids du fonctionnement de JEDI a reposé, et repose encore sur le pauvre "Secrétaire" de service aidé ou non par un nombre trop petit de bonnes volontés. Et c'est sans doute de cette centralisation excessive que périra JEDI si elle ne s'en sort pas. C'est d'autant plus dommage que je crois qu'elle correspondait à un besoin réel.

Qu'aux dieux ne plaise que mes craintes ne se réalisent! Et si une chance subsiste que JEDI redémarre, je pense qu'il faudrait prévoir des tâches bien précises et décentralisées, tenant compte de la dispersion des adhérents.

Mais peut-être l'association fonctionne-t-elle déjà comme cela et peut-être fais-je complètement erreur dans ce qui n'est après tout qu'un faisceau d'impressions.

Si JEDI disparait, je regretterai cette petite revue originale qui m'a apporté des lumières sur bien des trucs que je ne connaissais pas.

Comme je regrette de n'avoir pas reçu à temps l'annonce de la réunion extraordinaire...

Encore tous mes voeux de bonne santé à notre association, joints à mes remerciements sincères à la toute petite équipe qui s'est dévouée pour faire vivre JEDI jusqu'à présent. Elle a su faire un travail étonnant, et de quoi que soit fait le futur, elle mérite de chaleureuses félicitations.

Avec cette poignée de bonnes pensées mêlées de quelques regrets, j'adresse à l'équipe motrice toutes mes amitiés et, que diable, l'expression ed mon solide désir que cet épisode ne soit qu'un épisode!

B.C.LAMBEY
34070 MONTPELLIER

REPONSE: Cher adhérent,

Que de questions dans votre propos. Si la convocation a été envoyée in-extremis, c'est parce qu'il me fallait fixer une date en fonction de la garde de mes enfants (21 mois et 7 mois): ça a été le 26. Les convocations ont été triées de manière à privilégier d'abord les adhérents de la région parisienne, convocation reçue en majorité dans les temps.

Pour information, une association loi 1901 doit obligatoirement faire une réunion annuelle pour réélire le bureau: PRESIDENT, SECRETAIRE, TRESORIER. Par expérience, les précédentes réunions n'ont rassemblé que très peu de personnes, la dernière, celle du 26 a atteint la présence record de 4 personnes. Il est tout à fait évident qu'une association comme JEDI avec des adhérents dispersés ne peut exiger la présence d'un quorum qui serait impossible à tenir. Les vraies décisions se prennent tout au long de l'année, et pour JEDI, ont été jusqu'à présent réduites à des détails de fonctionnement: donner la signature bancaire au Secrétaire sur le chéquier de l'Association, investir dans la participation de l'édition du manuel F83, rechercher des sources de financement autres que les cotisations.

Pour le reste, bien que JEDI ait le statut d'association, c'est avant tout une revue, et de ce point de vue, son contenu a toujours été le reflet de la décentralisation même, celle-ci étant même étendue à des rédacteurs non adhérents. JEDI coûte environ 5000 Fr par mois et atteint 200 lecteurs. Existe-t-il un autre média dont on peut dire qu'il revient moins cher en investissement qu'un employé payé au SMIC (charges sociales comprises...)? Nous vous le disons, pour l'avoir entendu dans les salles de rédactions des grandes revues, que JEDI est envié, car nous abordons des sujets parfois difficiles, très spécifiques, ceci sans aucun budget publicitaire (mais aussi sans rétribuer les piges). Nous avons tenu jusqu'à présent plus longtemps que bien des revues (MICROTOM, MICROSTRAD, MICRODORE, TEOPHILE, MICRO7, etc...) mais nous ressentons tout de même les effets de l'effondrement du phénomène "MICRO".

Concernant la décentralisation telle que vous l'entendez, c'est à dire une décentralisation administrative, elle n'est pas interdite aux membres de JEDI d'en prendre l'initiative. Mais je me permet une remarque: plusieurs adhérents ont tenté de constituer leur

propre groupe d'activité FORTH, dont une certaine personne (qui se reconnaîtra) dont les coordonnées figuraient dans FORTH DIMENSION. Y a-t-il eu des contacts suivis après cette annonce? Pour ma part, mes coordonnées personnelles diffusées dans VIERTE DIMENSION (F.I.G. HAMBOURG RFA) n'ont eu aucun résultat. Pire, en écrivant aux coordonnées des différents groupes soi-disant constitués en Europe, je n'ai eu AUCUNE REPONSE (je tiens la liste à la disposition de qui la souhaite), sauf de la part de FIG HAMBOURG (encore eux... c'est une invasion teutonique).

Il y a de tout comme adhérents de JEDI: une banque, l'armée, l'éducation nationale, des industriels, des universitaires, des étudiants, des fonctionnaires, des chercheurs, des retraités, des curieux, des bidouilleurs, des débordés et des oisifs, ceux qui s'investissent et ceux qui investissent, ceux qui sont envahis par la micro et ceux qui envahissent les autres avec leur micro, ceux qui calculent, ceux qui écrivent, ceux qui dessinent, ceux qui musiquent (du verbe musiquer... ne cherchez pas, ça vient de sortir). De toutes ces compétences JEDI se veut le carrefour, un point de rencontre. Les grecs avaient la place publique, nous avons une revue et la liberté et les moyens de la diffuser: profitez-en!

LE SECRETAIRE

Dans le précédent article, nous avons abordé les principes de la logique et son application aux systèmes experts. Cette fois-ci, voyons le coté mathématique de la logique.

LA LOGIQUE BIEN FORMALISEE

Soient deux propositions:

"JEDI est une revue intéressante" nommée p
"JEDI est une association loi 1901" nommée q

on peut émettre une proposition globale regroupant nos deux propositions en une seule en disant:

"JEDI est une revue intéressante"
ET "JEDI est une association loi 1901"

Le terme ET s'appelle une **conjonction** (conjonction mathématique et non grammaticale):

$p \wedge q$ qui se lit " p ET q "

Nous pouvons disposer aussi des propositions suivantes:

"JEDI est le titre d'une revue"
OU "JEDI est un héros de film de fiction"

Il peut être l'un ou l'autre ou les deux (ce qui est le cas...):

$p \vee q$ qui se lit " p OU q "

Le terme OU s'appelle une **disjonction**.

Chaque proposition peut également avoir son contraire:

"JEDI est une revue"
"JEDI -n'est pas- une revue"

Exemple:

"SI JEDI est une revue"
"ALORS JEDI -n'est pas- un héros de film de fiction"

Le contraire ou **négation** se marque ainsi:

$\sim q$

Les combinaisons réalisées à partir des propositions ou de leur contraire à partir des conjonctions ou disjonctions sont appelées **prédicats**. Plus généralement, toute proposition peut être impliquée par d'autres propositions et leurs conjonction-disjontion-négation:

$p \vee q \vee r \equiv s$

Le signe ≡ marque l'implication.

La logique, comme l'algèbre, a ses postulats:

- toute combinaison débouchant sur une proposition systématiquement vraie est appelée **tautologie**:

$p \vee \sim p \equiv s$

- toute combinaison débouchant sur une proposition systématiquement fausse est une **contradiction**:

$p \wedge \sim p \equiv s$

Voilà, vous disposez de TOUS les éléments pour travailler et étudier la logique et le calcul des prédicats.

LA NOTATION

En logique, la notation n'est pas normalisée. Cependant, en fonction du domaine d'utilisation, une habitude typographique prédomine.
Notation mathématique:

ET ^
OU v
NON ~

Notation en automatisme industriel:

ET . ou rien
OU +
NON barre au-dessus de la variable ou de l'expression

On trouve également dans certains ouvrages les formulations suivantes:

ET &
NON ¬

Dans chacune de ces notations, les parenthèses sont admises. Dans ce cas, il faut tenir compte de l'ordre de priorité des opérateurs logiques ET et OU qui sont identiques aux ordres de priorité des opérateurs arithmétique * et +.

Notation des équivalences:

≡ en mathématique
= en automatique industriel

Enoncé de condition:

→ en mathématique

Exemple:

Si a ET non B alors q

est noté

$a \wedge \sim b \rightarrow q$

LES LOIS DE LA LOGIQUE OU DE L'ALGEBRE DES PROPOSITIONS

Loi idempotentes:

$p \vee p \equiv p$
$p \wedge p \equiv p$

Loi d'associativité: $(p \vee q) \vee r \equiv p \vee (q \vee r)$

Loi de commutativité: $p \vee q \equiv q \vee p$

Loi de distributivité: $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

La plupart de ces lois ont un comportement identique à celles définies en arithmétique, à la différence près que si vous notez en automatisme industriel cette expression

$a + b = L$ où $a = 1$ et $b = 1$

L ne vaudra pas 2 en décimal ou 10 en binaire, mais 1 en logique! Ne perdez pas de vue que la logique formelle nie l'aspect sémantique des expressions équivalentes exprimées dans un langage plus évolué. Si vous dites:

SI j'ai une fraise ALORS je peux repeindre le plafond

se formule $a \rightarrow q$

Un système de résolution logique traitera les expressions sous la seconde forme, la première forme devant être d'abord traduite. C'est ici qu'intervient le moteur d'inférence

LE RAISONNEMENT LOGIQUE

Le raisonnement est la relation établie entre diverses propositions et une conclusion. La conclusion peut être elle même proposition d'une autre conclusion:

$P1,P2,...,Pn-1,Pn \vdash Q$

Le raisonnement Q est valide si P1,P2,...,Pn sont valides. Exemple de formulation dans un système expert:

SI a ET SI b OU SI c ALORS q

noté également

SI a ET b OU c ALORS q

Cas de syllogisme:

$p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$

SI p ALORS q
SI q ALORS r

SI un homme est célibataire
ALORS il est malheureux
SI un homme est malheureux
ALORS il meurt jeune

ce qui peut amener à la conclusion:

SI un homme est célibataire
ALORS il meurt jeune

ce qui prouve l'utilité des belles-méres, car on entend toujours l'expression: "ma belle-mère, ça fait trente ans que je la supporte...", donc ceux qui meurent jeune ne sont pas mariés, donc il n'y aura plus de mariage, car les jeunes ne vivent pas assez longtemps pour se marier, ce qui risque d'être la fin des belles-méres...

TABLES DE VERITES ET EXPLOSION COMBINATOIRE

Dans la réalité, et le dernier exemple le démontre clairement, une conclusion peut dépendre de plus d'un ou deux paramètres, eux-même dépendants d'autres paramètres. Heureusement, sinon nous serions contraints à nous marier très jeune pour survivre...

Afin de ne laisser aucune possibilité de coté, on utilise des tables de vérité. A gauche figurent les prémisses ou propositions, à droite figurent les conditions résultant des relations entre les propositions. Exemple:

a	b	avb
F	F	F
F	V	V
V	F	V
V	V	V

Ici, nous ne disposons que de deux propositions, a et b. Avec trois propositions, nous aurions trois colonnes et huit lignes dans notre table de vérité. Le nombre de colonnes à réserver dans une table de vérité est égal au nombre de propositions, le nombre de lignes est égal à 2 élevé à la puissance du nombre de propositions. Pour 16 propositions, nous aurions une table de vérité de 16 colonnes et de 65536 lignes.

En automatisme industriel, les séquences logiques entre deux lignes d'une table de vérité ne doivent pas contenir plus d'une variable changeant de valeur simultanément. On réécrit donc le tableau:

a	b	avb
F	F	F
F	V	V
V	V	V
V	F	V

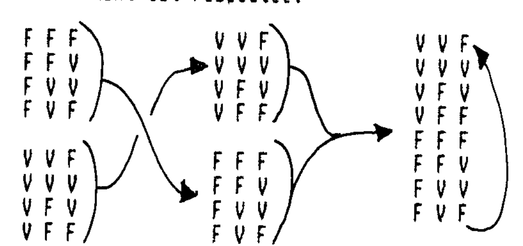
Cette disposition a une propriété particulière, elle peut être bouclée:



Avec trois variables, il faut définir les séquences logiques:

	1er bouclage	seconds bouclages
F F F	F F F	F F F
F F V	F F V	F F V
F V V	F V V	F V V
F V F	F V F	F V F
V V F	V V F	V V F
V V V	V V V	V V V
V F V	V F V	V F V
V F F	V F F	V F F

Dans des boucles isolées, les sous-boucles sont commutatives, la condition selon laquelle il ne doit pas y avoir plus d'une proposition changeant d'état simultanément est respectée:



Cette commutativité reste valable quelque soit le nombre de variables. Voilà un aspect intéressant de la logique: quelles sont les règles permettant la génération automatique des séquences en respectant la commutativité des boucles? Si vous trouvez, faites-le savoir! (pour vous aider, cherchez dans la théorie des anneaux commutatifs).

à suivre...